JOURNAL OF
Emerging Paradigms
in Computing Systems
JEPCS

Journal of Emerging Paradigms in Computing Systems (JEPCS) | Volume 01, Issue No. 01, 2025

ARTICLE TYPE: Research Article

Article ID: JEPCS-2025-0101-0011

An Enhanced Hybrid PSO-LSTM Framework for Energy-Aware Task Scheduling in Heterogeneous Cloud Computing Environments

Zain ul Huda[1,*]

[1] Department of Computer Science, University of Central Punjab, Lahore, Pakistan

* Corresponding author: L1F20BSCS0346@ucp.edu.pk

Received: September 24, 2024 **Accepted:** December 22, 2024

Abstract- Cloud task scheduling is an NP-hard optimization problem in which submitted tasks must be mapped to virtual machines under competing performance objectives. Particle Swarm Optimization (PSO) is widely used for this problem because of its simple update procedure, but the standard algorithm can converge prematurely. In the selected PSO scheduling papers, task sets are also fixed before optimization and next-window workload prediction is not incorporated into the particle update. This paper presents a Hybrid PSO-LSTM framework for energy-aware scheduling in heterogeneous cloud environments. A Long Short-Term Memory (LSTM) model learns historical task-arrival sequences and estimates the workload for the next scheduling window. The prediction stage provides an additional guidance term for PSO, while candidate task-to-VM assignments are evaluated using makespan, energy consumption, and resource utilization. The framework is implemented in CloudSim and compared with standard PSO, Genetic Algorithm, Round Robin, First-Come, First-Served, and Whale Optimization Algorithm using workloads of up to 1000 tasks and 50 virtual machines. Across the reported comparisons, PSO-LSTM records lower makespan and energy consumption and higher resource utilization and throughput. The makespan-based scalability experiment shows the same relative trend within the simulated workload range.

Keywords- Cloud computing; Task scheduling; Particle Swarm Optimization; Long Short-Term Memory; Energy-aware scheduling

1. Introduction

Cloud computing provides virtualized processing, storage, and networking resources that can be provisioned according to demand. In this environment, task scheduling determines how submitted jobs are assigned to the available virtual machines (VMs). The assignment affects completion time, resource utilization, and the operating cost of the cloud platform [1-2], [6], [17].

Cloud task scheduling is commonly modeled as an optimization problem in which a set of tasks must be mapped to a set of heterogeneous VMs. The problem is NP-hard, so exhaustive search becomes impractical as the number of possible assignments grows [6]. Metaheuristic methods, including Particle Swarm Optimization (PSO), are therefore widely used to search for feasible schedules within reasonable computation time [2], [9].

PSO has a simple update rule, but the standard algorithm can lose population diversity and converge prematurely [9]. In addition, several PSO-based cloud schedulers are evaluated on task sets that are fixed before optimization begins [1], [2], [6]. Such formulations do not use an estimate of the workload expected in the next scheduling window. Energy is also not included in every PSO scheduling objective [1]-[4], although energy-aware PSO schedulers have been reported [6]. Multi-objective scheduling is likewise well established in the literature [6]-[8], [17]. The distinction pursued in this

paper is therefore not multi-objective scheduling alone; it is the use of workload prediction as an additional PSO guidance signal while evaluating makespan, energy consumption, and resource utilization together.

This paper presents a Hybrid PSO-LSTM framework for energy-aware task scheduling in a heterogeneous cloud environment. The LSTM component learns historical task-arrival patterns and predicts the workload for the next scheduling window. The enhanced PSO component uses prediction-derived guidance during particle updates. Candidate task-to-VM assignments are evaluated using makespan, energy consumption, and resource utilization.

The main contributions are as follows:

- We couple LSTM-based workload prediction with PSO-based task scheduling so that anticipated workload conditions can influence the particle search.
- We introduce a PSO guidance term derived from the workload-prediction stage and compare the resulting scheduler with standard PSO and four additional baselines.
- The scheduling objective evaluates makespan, energy consumption, and resource utilization together. This combination differs from the objectives used by the selected comparison papers, although multi-objective cloud scheduling itself is not new.
- The framework is implemented in CloudSim and evaluated with workloads of 100, 200, 500, and 1000 tasks over 10, 20, 30, and 50 VMs.
- The reported experiments include makespan, energy consumption, cost, throughput, convergence, response time, load distribution, memory use, and scalability measures.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the proposed Hybrid PSO-LSTM framework. Section 4 examines the selected base PSO paper and identifies the limitations that are relevant to the proposed framework. Section 5 shows the results. Section 6 concludes the paper with the future work.

2. Related Work

For scheduling, Anbarkhan and Rakrouki [1] developed an Enhanced PSO (EPSO) method that changes the way particles are initialized and adapts the search procedure during optimization. Their test considers both workflow and independent task sets, with the largest experiment containing 300 tasks. Nabi et al. [2] followed a different PSO strategy where the inertia weight is adjusted as the search progresses. They evaluate the results using average resource utilization ratio (ARUR), throughput and makespan. Wang and Zuo [3] considered deadline-constrained workflow scheduling instead of optimization of task set. They combine PSO with idle-time slot aware rules and minimizes workflow execution cost. Shaheen and Santhosh Kumar [4] also use PSO, but within the PSO-BATS (bandwidth-aware divisible task scheduling). Their approach includes load-balancing and host-employment mechanisms. In another paper, the Rjoub et al. [5] examined reinforcement learning, deep Q networks, RNN-LSTM, and DRL-LSTM for task scheduling in large-scale cloud systems. They use real cloud traces, with task waiting time and resource-consumption cost to evaluate the scheduling decisions. This work is different from PSO-based methods because the scheduling policy is learned rather than obtained through particle-based search. Other papers have modified the optimization rules which are required during scheduling. Alsaidy et al. [6] improve the initial PSO population using the longest-job-to-fastest-processor and minimum-completion-time heuristics. Their experiments report makespan, total execution time, degree of imbalance, and total energy consumption. Hamed and Alkinani [7] use a genetic algorithm and formulate the scheduling problem around three objectives: completion time, execution cost, and resource utilization. Jia et al. [8] approach the same class of problem with an improved Whale Optimization Algorithm. Their formulation combines execution time, cost, and VM load after the individual objective terms are normalized. The behavior of PSO itself has also been examined independently of a particular cloud scheduler. Shami et al. [9] reviewed PSO variants, applications, and limitations, and identified premature convergence as a recurring problem when particle diversity is reduced during the search. Bi et al. [10], by contrast, paper prediction rather than task assignment. Their model combines multi-head attention with LSTM variants to forecast cloud workload and resource demand. The predicted quantities are useful for anticipating future resource requirements, but the paper does not perform task-to-VM scheduling. They combined multi-head attention with LSTM variants to forecast workload and resource usage in cloud data centers. The paper addresses prediction rather than task assignment. Xing [11] proposed a DRL-LSTM scheduling approach that also incorporates Deep-Q-Network components for workload planning in edge/cloud environments. The reported measures include service time, VM utilization, and failed-task rate. Islam et al. [12] used deep reinforcement learning for Spark job scheduling and jointly considered performance and execution cost in cloud environments. Katal et al. [13] surveyed software-level approaches for energy-efficient cloud data centers and reported data-center electricity-consumption trends as part of the motivation for energy-aware resource management. Wang et al. [14] formulated security-aware task scheduling with deadline constraints for heterogeneous hybrid clouds and optimized task completion and public-resource cost under security requirements. Ouhame et al. [15] developed a CNN-LSTM model for forecasting CPU, memory, and network utilization in cloud data centers. Dogani et al. [16] combined CNN, GRU, and attention mechanisms for multivariate workload and resource prediction using cloud traces. Moazeni et al. [17] proposed adaptive multi-objective teaching-learning-based optimization for dynamic cloud resource allocation. Their objectives include makespan, cost, and resource utilization. Alahmad et al. [18] developed a proactive

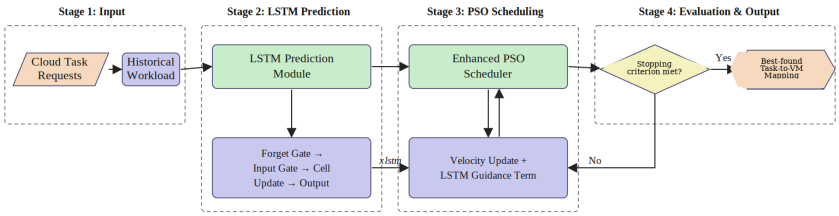


Figure 1: Processing stages of the Hybrid PSO-LSTM task scheduler.

failure-aware task-scheduling framework that predicts task failures and selects remedial scheduling actions. Khan et al. [19] reviewed machine-learning-centric resource management in cloud computing, including prediction, allocation, and scheduling directions. CloudSim, used for the experiments in this paper, was introduced by Calheiros et al. [20] as a toolkit for modeling and simulating cloud-computing environments and resource-provisioning policies.

2.1. Gap Analysis

The literature does not support a claim that existing cloud schedulers are uniformly single-objective or uniformly energy-unaware. For example, Alsaidy et al. [6] explicitly evaluate energy consumption, while Hamed and Alkinani [7], Jia et al. [8], and Moazeni et al. [17] consider multiple objectives. The gap addressed here is narrower. The selected PSO papers do not combine next-window workload prediction with an LSTM-derived guidance term inside the PSO update while jointly evaluating makespan, energy consumption, and resource utilization.

Three observations motivate this design. First, the PSO scheduling papers in [1], [2], and [6] operate on task sets available to the scheduler rather than predicting the next task-arrival window. Second, premature convergence remains a recognized PSO limitation [9]. Third, workload-prediction papers such as [10] forecast future demand but do not perform PSO-based task assignment. Learning-based schedulers such as [5] and [11] integrate temporal models with scheduling, but they use reinforcement-learning-based decision mechanisms rather than prediction-guided PSO.

Table 1 summarizes the scope of the closest papers and the specific distinction of the proposed framework.

Table 1: Scope of selected task-scheduling and workload-prediction papers relative to this work.

Ref.	Approach	Main objective	Prediction model	Energy	Relation to this work
[1]	EPSO	Makespan; scheduling cost evaluated	No	No	PSO scheduling without workload prediction
[2]	AdPSO	Makespan, throughput, ARUR	No	No	Adaptive PSO without next-window forecasting
[3]	PSO + idle-slot rules	Execution cost under a deadline	No	No	Workflow scheduling without prediction-guided PSO
[4]	PSO-BATS + MLRHE	Cost, PIR, makespan	No	No	Load-balanced PSO scheduling without LSTM forecasting
[5]	RL / DQN / LSTM / DRL-LSTM	Waiting time and resource-consumption cost	RNN/LSTM	No	Learning scheduler rather than PSO guidance
[6]	Heuristic-initialized PSO	Makespan, execution time, imbalance, energy	No	Yes	Energy-aware PSO, but no workload forecasting
[7]	GA	Completion time, execution cost, utilization	No	No	Multi-objective scheduling without prediction
[8]	Improved WOA	Time, cost, VM load	No	No	Multi-objective scheduling without prediction
[10]	Hybrid LSTM + attention	Workload and resource prediction	Yes	N/A	Prediction model without task scheduler
[11]	DRL-LSTM / DQN	Service time, VM utilization, failed-task rate	LSTM	No	Learning-based scheduling without PSO

3. Proposed Methodology

The proposed scheduling method is organized into four parts. The cloud system model defines the tasks and heterogeneous VMs, the LSTM estimates the workload for the next window, and the enhanced PSO uses that estimate during particle search. Candidate schedules are finally scored by the multi-objective fitness function. The implementation settings used for evaluation are given after the method description.

3.1. System Architecture

The complete processing sequence is given in Figure 1. Task requests and historical workload records form the input. The historical sequence is processed by the LSTM, and its predicted workload is supplied to the enhanced PSO scheduler. PSO then searches candidate task-to-VM mappings until the stopping condition is reached, after which the selected mapping is returned.

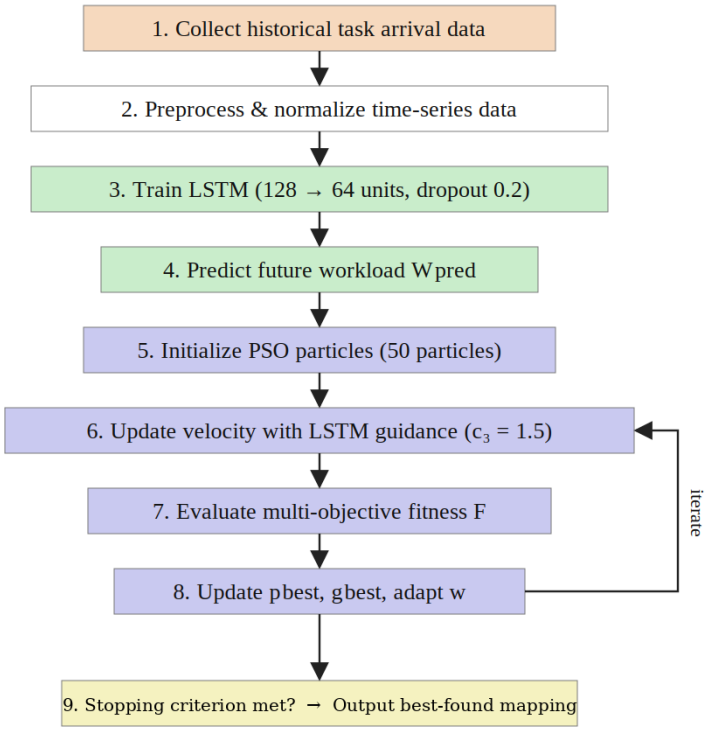


Figure 2: Data flow process of the Hybrid PSO-LSTM scheduling process.

3.2. LSTM-PSO Data Flow

Figure 2 gives a step-by-step view of the information passed between the two modules. Historical arrivals are collected and normalized before LSTM training. The predicted workload is then used during PSO initialization and velocity updating, followed by fitness calculation, pbest/gbest updates, and the stopping-condition check.

3.3. System Model

The scheduler considers a heterogeneous cloud with task set $T = \{t_1, t_2, \ldots, t_n\}$ and VM set $V = \{v_1, v_2, \ldots, v_m\}$. Here, n and m denote the numbers of submitted tasks and available VMs, respectively. Each task is represented by its computational length in Million Instructions (MI). Each VM is characterized by its processing rate in Million Instructions Per Second (MIPS) and by the power value used in the energy calculation.

3.4. LSTM Workload Prediction Module

The LSTM module predicts the workload of the next scheduling window from a historical task-arrival sequence. The reported network contains two LSTM layers with 128 and 64 hidden units, a dropout rate of 0.2, and a dense output layer.

The standard LSTM updates are given in Equations 1-5:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \tag{1}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \tag{2}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right), \tag{3}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right), \tag{4}$$

$$h_t = o_t \odot \tanh(C_t). \tag{5}$$

Here, σ is the sigmoid activation function, x_t is the input at time t, h_t is the hidden state, C_t is the cell state, W denotes weight matrices, and b denotes bias vectors.

Algorithm 1 Hybrid PSO-LSTM Framework

Require: Task set T, VM set V, historical workload data H
Ensure: Best task-to-VM mapping found
Train the LSTM on historical data H
Predict next-window workload W_{pred}
Construct guidance vector x_{lstm} from W_{pred} using the particle encoding
Initialize N particles and evaluate their fitness
Set each $pbest_i$ to its initial particle position and set $gbest$ to the best initial particle
for $iter = 1$ to $MaxIter$ **do**
  for each particle i **do**
    Update v_i using Equation 8
    Update x_i using Equation 7 and enforce the task-to-VM encoding constraints
    Calculate fitness $F(x_i)$ using Equation 12
    if $F(x_i) < F(pbest_i)$ **then**
      Update $pbest_i$
    end if
    if $F(x_i) < F(gbest)$ **then**
      Update $gbest$
    end if
  end for
  Update adaptive inertia weight w
end for
Assign tasks to VMs according to $gbest$
return Best task-to-VM mapping found

3.5. Enhanced PSO Scheduling Module

The PSO population contains 50 particles and runs for at most 100 iterations. The inertia weight w decreases from 0.9 to 0.4, while the cognitive and social coefficients are $c_1 = 2.0$ and $c_2 = 2.0$, respectively. Each particle position x_i represents a candidate task-to-VM mapping.

For standard PSO, velocity and position are updated by Equations 6 and 7:

$$v_i(t+1) = w\, v_i(t) + c_1 r_1 \left(pbest_i - x_i(t)\right) + c_2 r_2 \left(gbest - x_i(t)\right), \tag{6}$$

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{7}$$

The proposed update introduces an additional guidance vector:

$$\begin{aligned} v_i(t+1) = w\, v_i(t) &+ c_1 r_1 \left(pbest_i - x_i(t)\right) \\ &+ c_2 r_2 \left(gbest - x_i(t)\right) + c_3 r_3 \left(x_{lstm} - x_i(t)\right). \end{aligned} \tag{8}$$

The coefficient $c_3 = 1.5$ controls the additional term, and r_1, r_2, and r_3 are random values in $[0, 1]$. The LSTM predicts the next-window workload W_{pred}, and the prediction stage provides the guidance vector x_{lstm} used in Equation 8.

3.6. Multi-Objective Fitness Function

Makespan, energy consumption, and resource utilization are defined in Equations 9-11, respectively:

$$MS = \max_i \{FT(t_i)\}, \tag{9}$$

$$E = \sum_{j=1}^{m} P_j\, ET(v_j), \tag{10}$$

$$RU = \frac{1}{m} \sum_{j=1}^{m} \frac{AT(v_j)}{TT(v_j)}. \tag{11}$$

Here, $FT(t_i)$ is the finish time of task t_i, P_j is the power consumption of VM v_j, $ET(v_j)$ is its execution time, and $AT(v_j)$ and $TT(v_j)$ are its active and total times. The manuscript gives the weighted fitness as

$$F = \alpha\, MS + \beta\, E - \gamma\, RU, \tag{12}$$

with $\alpha = 0.4$, $\beta = 0.4$, $\gamma = 0.2$, and $\alpha + \beta + \gamma = 1.0$.

Algorithm 1 summarizes the prediction and scheduling procedure. The term "best-found" is used rather than "optimal" because a metaheuristic does not guarantee a global optimum.

4. Critical Analysis of Base Work

This section examines the base paper by Anbarkhan and Rakrouki [1]. The comparison is restricted to properties that can be verified from that paper and that are relevant to the present PSO-LSTM design. The base paper proposes an Enhanced PSO (EPSO) method for workflow-task scheduling. It reduces particle dimensions through workflow-task preprocessing, modifies particle initialization, and introduces a self-adaptive search function. The CloudSim experiments compare EPSO with HPSO and standard PSO. Workflow and independent task sets containing 10, 20, 50, 100, 150, 200, 250, and 300 tasks are evaluated, and the reported results include scheduling time and cost [1].

4.1. Limitations Relevant to the Present Work

- The base experiments vary the number and type of tasks, but the task sets are prepared for each simulation run. No next-window workload predictor is connected to the PSO search. The present paper therefore differs by introducing a workload-prediction stage before scheduling.
- The base paper evaluates workflow scheduling time and cost, but energy consumption is not part of its objective formulation or reported comparison [1]. Energy-aware PSO scheduling does exist elsewhere in the literature [6], so this point should be interpreted as a difference from the selected base paper rather than a general absence in prior work.
- EPSO modifies the search process and reports improved convergence, but PSO in general remains susceptible to premature convergence when population diversity is lost [9]. The present paper attempts to influence the search with prediction-derived guidance; whether this term is responsible for improved convergence should ideally be tested through an ablation paper.
- The base formulation minimizes makespan and separately evaluates scheduling cost [1]. The proposed method instead reports a weighted objective containing makespan, energy consumption, and resource utilization. This is a difference in objective design, not evidence that multi-objective cloud scheduling is new; multi-objective schedulers already exist [6]-[8], [17].
- The base paper evaluates up to 300 tasks [1], whereas the present manuscript reports up to 1000 tasks. This provides a larger task-count experiment relative to the selected base paper, but it should not be described as production-scale validation.

4.2. Justification for Proposed Work

The proposed framework extends the selected base design by adding workload prediction and by including energy consumption and resource utilization in the scheduling objective.

The present experiments extend the task-count range used in the selected base paper. This provides a broader workload comparison, but the findings remain tied to the CloudSim configurations examined here. Stronger claims about convergence behavior, scalability, or general superiority would require repeated independent runs, an ablation analysis of the prediction component, and evaluation at a larger deployment scale.

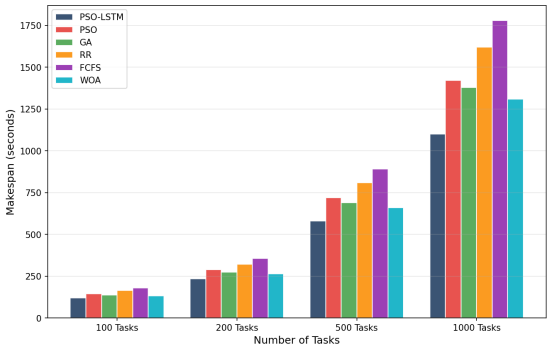
5. Results and Discussion

The implementation and simulation parameters used in the experiments are shown in Table 2.

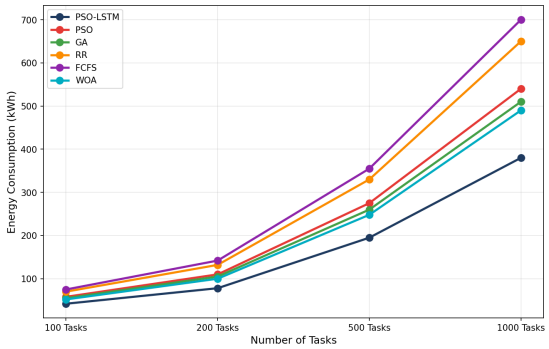
The performance of the Hybrid PSO-LSTM scheduler is examined in CloudSim from several perspectives. Rather than presenting each result separately, the 15 plots are arranged into four groups according to the aspect of the scheduler being evaluated.

The first group considers the main scheduling objectives. Makespan for 100, 200, 500, and 1000 tasks is reported in Figure 3(a). PSO-LSTM gives the lowest makespan at all four workload sizes. The separation from the other methods is modest at the smaller task counts but becomes more pronounced at 500 and 1000 tasks. Within the evaluated range, the increase in workload does not change the relative ordering observed for PSO-LSTM.

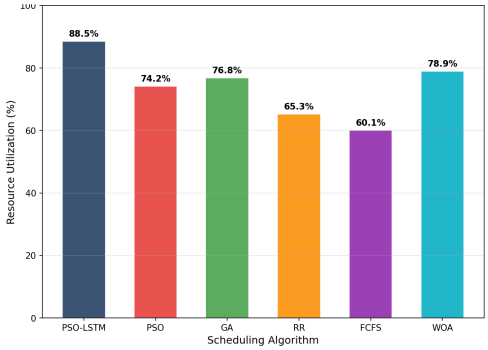
Energy consumption is reported in Figure 3(b). Energy demand rises as the number of submitted tasks increases, which is expected because more work must be processed by the available VMs. The PSO-LSTM curve remains below the other five schedulers at each workload, giving the lowest reported energy consumption in these experiments.



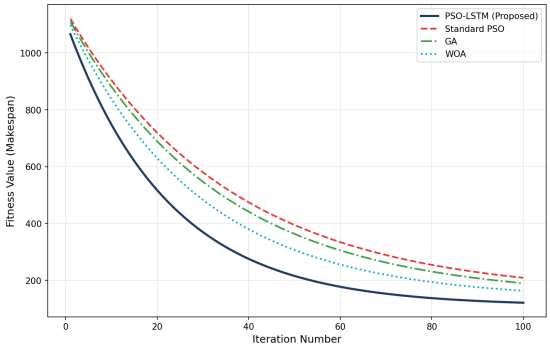
(a) Makespan for the evaluated task sizes.



(b) Energy consumption with increasing workload.



(c) Resource utilization of the schedulers.



(d) Convergence over 100 iterations.

Figure 3: Core scheduling results for makespan, energy consumption, resource utilization, and convergence.

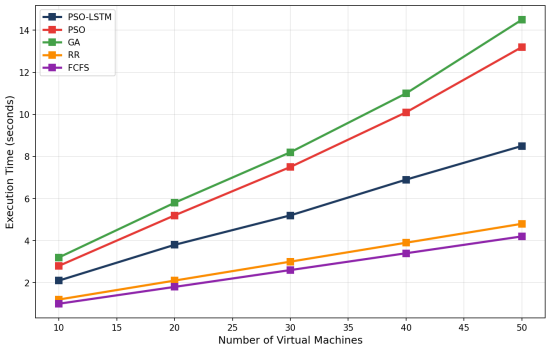
The utilization results in Figure 3(c) provide a different view of resource use. PSO-LSTM reaches 88.5%, followed by WOA at 78.9%, GA at 76.8%, and standard PSO at 74.2%. RR and FCFS record 65.3% and 60.1%, respectively. The proposed scheduler therefore makes greater use of the available VM capacity in this comparison.

Figure 3(d) shows search over 100 iterations. The PSO-LSTM curve falls more quickly during the earlier part of the run and ends below the standard PSO, GA, and WOA curves. This pattern is consistent with faster convergence in the reported experiment. It does not, however, establish that the LSTM guidance term is solely responsible for the difference. An ablation study would be needed to separate its effect from the other components of the scheduler. This result describes the convergence behavior observed in the reported experiment. An ablation analysis would be required to determine how much of the improvement is attributable specifically to the LSTM guidance term.

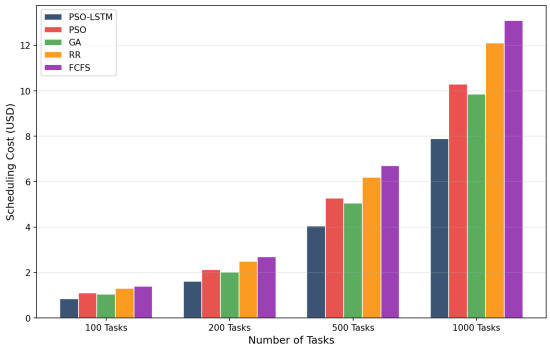
Execution time for different numbers of available VMs is reported in Figure 4(a). The PSO-LSTM curve remains below standard PSO and GA throughout the evaluated VM range. RR and FCFS, however, report lower execution times than the metaheuristic methods in this particular experiment. Execution time alone therefore does not represent the complete scheduling trade-off and should be considered together with the other reported metrics.

Table 2: CloudSim implementation and simulation configuration.

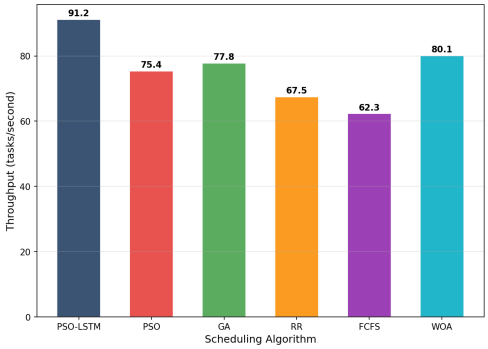
Parameter	Configuration
Simulation Toolkit	CloudSim 3.0
Implementation Language	Java
Processor	Intel Core i7
Memory	16 GB RAM
Number of VMs	10, 20, 30, and 50
Number of Tasks	100, 200, 500, and 1000
VM Processing Speed	1000–3000 MIPS
Task Length	1000–50000 MI
Scheduling Policy	Time-shared
Baseline Algorithms	Standard PSO, Genetic Algorithm (GA), Round Robin (RR), First-Come, First-Served (FCFS), and Whale Optimization Algorithm (WOA)



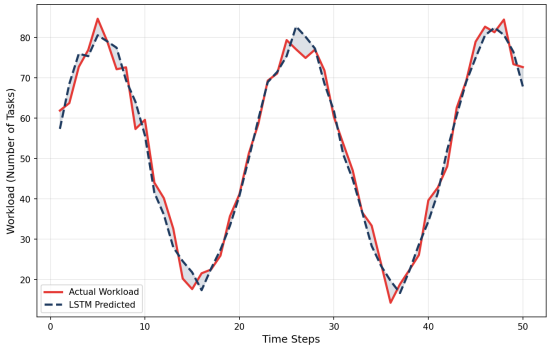
(a) Execution time versus the number of VMs.



(b) Scheduling cost for increasing workloads.



(c) Throughput of the compared schedulers.



(d) Actual and LSTM-predicted workload.

Figure 4: Operational and prediction results: execution time, scheduling cost, throughput, and workload prediction.

Scheduling cost is given in Figure 4(b). The cost increases with task count for every scheduler. PSO-LSTM records the lowest plotted cost at 100, 200, 500, and 1000 tasks. The difference between the methods also becomes larger at the higher workloads.

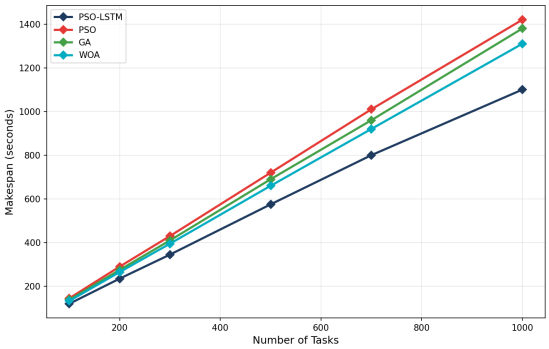
Throughput is compared in Figure 4(c). PSO-LSTM reaches 91.2 tasks/s. WOA and GA record 80.1 and 77.8 tasks/s, respectively, while standard PSO reaches 75.4 tasks/s. RR records 67.5 tasks/s and FCFS 62.3 tasks/s. Among the compared methods, PSO-LSTM therefore provides the highest throughput in the reported experiment.

The workload-prediction trace appears in Figure 4(d). The predicted sequence follows the main rises and falls of the actual task-arrival pattern. Differences are still visible around several local peaks and troughs.

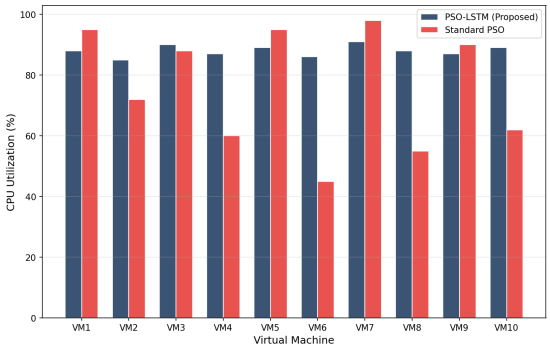
Figure 5(a) shows how makespan changes as the workload grows to 1000 tasks. All four methods require more time as additional tasks are introduced, but PSO-LSTM remains below standard PSO, GA, and WOA throughout the plotted range. The result shows that its makespan advantage is maintained for the workloads tested here. Since the experiment stops at 1000 tasks and is carried out in CloudSim, it does not by itself establish scalability in a production cloud environment.

Figure 5(b) shows the CPU-utilization behaviour across different VMs. Utilization under PSO-LSTM is high within a relatively narrow range across the ten VMs. Standard PSO varies more widely, with several VMs operating at noticeably lower utilization. For this workload, the smaller variation suggests that PSO-LSTM distributes processing activity more evenly across the available VM pool.

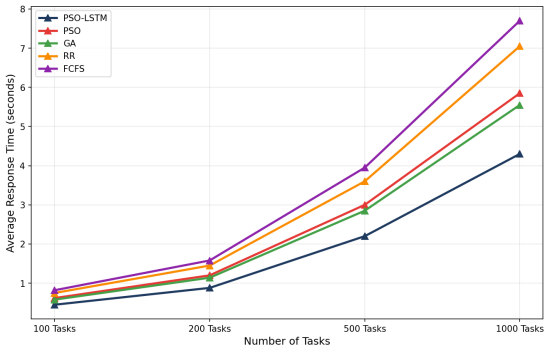
Average response time is shown in Figure 5(c). Increasing the number of tasks from 100 to 1000 raises the response time of every scheduler. PSO-LSTM remains the lowest curve at each workload. The difference is limited at the smaller task sets but becomes clearer at 500 and 1000 tasks. Scheduling outcomes are separated into successful, partial, and failed cases in Figure 5(d). PSO-LSTM completes 95.5% of the reported schedules successfully, while 3.2% are partial and 1.3% fail. Standard PSO records 82.3% successful schedules, with 11.4% partial and 6.3% failed cases. Within this experiment, PSO-LSTM therefore produces a larger share of successful scheduling outcomes and smaller shares of partial and failed cases. In Figure 6(a) the CPU behaviour over the simulation time is shown. The PSO-LSTM trace remains in a comparatively high-utilization region for most of the run and changes within a narrower band. Standard PSO shows larger variations, including several periods in which CPU utilization falls more sharply. The result is consistent with the more even VM-level utilization observed in the preceding experiment. Memory requirements are compared in Figure 6(b). PSO-LSTM uses 42.5 MB, the smallest value reported in the plot. FCFS and RR follow at 45.8 MB and 48.2 MB. Higher values are recorded for WOA at 54.7 MB, GA at 56.1 MB, and standard PSO at 58.3 MB. The difference indicates that the additional LSTM-guided scheduling stage does not result in the largest memory footprint among the evaluated methods under this configuration.



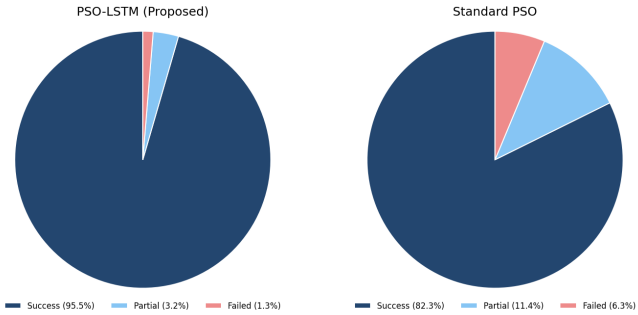
(a) Makespan-based scalability.



(b) CPU-utilization distribution across VMs.

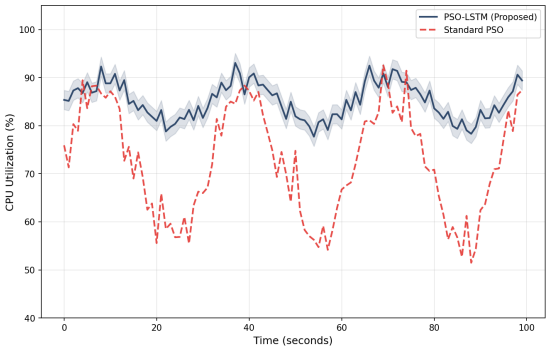


(c) Average response time.

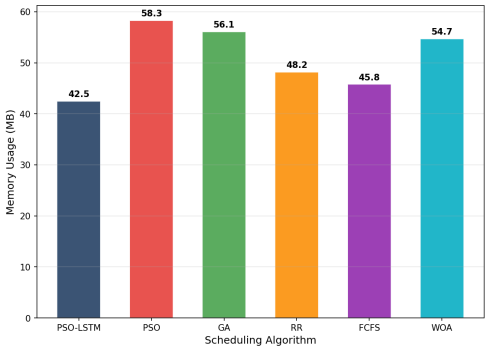


(d) Successful, partial, and failed schedules.

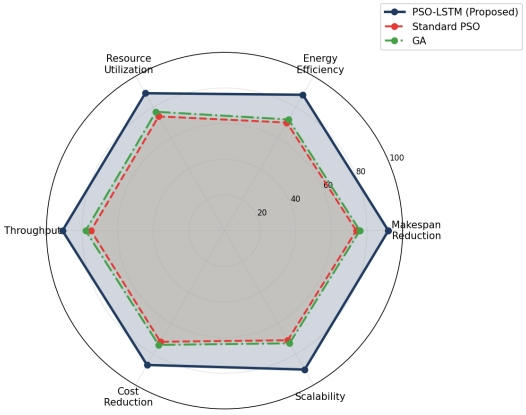
Figure 5: Scalability and scheduling behavior in terms of makespan, load distribution, response time, and scheduling outcomes.



(a) CPU utilization over the simulation interval.



(b) Memory footprint of the schedulers.



(c) Normalized overall performance.

Figure 6: Resource behavior and overall comparison based on CPU utilization, memory usage, and normalized performance measures.

The radar plot in Figure 6(c) brings together normalized values for makespan reduction, energy efficiency, resource utilization, throughput, cost reduction, and scalability. Only PSO-LSTM, standard PSO, and GA are included in this comparison. PSO-LSTM extends farther on each of the six axes than the other two methods. Because these values are normalized, the radar plot is best read as a visual summary of the trends already reported for the individual metrics rather than as a separate performance result. Taken as a whole, the CloudSim experiments show a consistent pattern in favor of PSO-LSTM across most of the reported measures. Lower values are obtained for makespan, energy consumption, scheduling cost, and response time in the corresponding comparisons, while resource utilization, throughput, and scheduling success rate are higher. The VM-level results also show a narrower utilization spread, and the CPU trace varies less than that of standard PSO during the reported interval.

These results remain specific to the workload sizes, They should therefore be read as results for the reported CloudSim experiments rather than as evidence of general superiority across cloud platforms.

6. Conclusion and Future Work

In this paper, a Hybrid PSO-LSTM scheduler is developed for heterogeneous cloud environments. Historical task arrivals are processed by the LSTM to estimate the workload of the next scheduling window. That estimate enters the PSO velocity update as an additional guidance term. Makespan, energy consumption, and resource utilization are then evaluated together in the fitness function. The design is intended to address the static-workload assumption, limited energy awareness, single-objective formulation, and premature-convergence issue discussed earlier. The implementation is evaluated in CloudSim against standard PSO, GA, RR, FCFS, and WOA with workloads of up to 1000 tasks and 50 VMs. For the simulated cases, PSO-LSTM records lower makespan, energy consumption, cost, and response time, together with higher resource utilization, throughput, scheduling success rate, and more even VM utilization than the compared baselines. These results are limited to the reported simulation settings.

Future work will test larger production-scale deployments and workloads with more dynamic and unpredictable task-arrival patterns.

Author Declarations

Author Contributions (CRediT): Zain ul Huda: Conceptualization, Methodology, Software, Investigation, Formal Analysis, Writing - Original Draft.

Funding: No external funding was received for this research.

Data Availability Statement: The experiments were carried out in CloudSim. The implementation details available from the author upon reasonable request.

Acknowledgments: No acknowledgments are reported.

Conflicts of Interest: The author reports no conflicts of interest.

Ethics Statement: Not applicable

Declaration of Generative AI: An AI-assisted tool was used during manuscript preparation for language editing and formatting support. The author checked the revised content and remains responsible for the accuracy, integrity, and content of the manuscript.

References

[1] S. H. Anbarkhan and M. A. Rakrouki, "An Enhanced PSO Algorithm for Scheduling Workflow Tasks in Cloud Computing," *Electronics*, vol. 12, no. 12, Art. no. 2580, 2023, doi: 10.3390/electronics12122580.

[2] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing," *Sensors*, vol. 22, no. 3, Art. no. 920, 2022, doi: 10.3390/s22030920.

[3] Y. Wang and X. Q. Zuo, "An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 5, pp. 1079-1094, May 2021, doi: 10.1109/JAS.2021.1003982.

[4] A. R. Shaheen and S. Santhosh Kumar, "Tasks Scheduling in Cloud Environment Using PSO-BATS with MLRHE," *Intell. Automat. Soft Comput.*, vol. 35, no. 3, pp. 2963-2978, 2023, doi: 10.32604/iasc.2023.025780.

[5] G. Rjoub, J. Bentahar, O. A. Wahab, and A. S. Bataineh, "Deep and Reinforcement Learning for Automated Task Scheduling in Large-Scale Cloud Computing Systems," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 23, Art. no. e5919, 2021, doi: 10.1002/cpe.5919.

[6] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic Initialization of PSO Task Scheduling Algorithm in Cloud Computing," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, Part A, pp. 2370-2382, 2022, doi: 10.1016/j.jksuci.2020.11.002.

[7] A. Y. Hamed and M. H. Alkinani, "Task Scheduling Optimization in Cloud Computing Based on Genetic Algorithms," *Comput. Mater. Contin.*, vol. 69, no. 3, pp. 3289-3301, 2021, doi: 10.32604/cmc.2021.018658.

[8] L. W. Jia, K. Li, and X. Shi, "Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm," *Wirel. Commun. Mob. Comput.*, vol. 2021, Art. no. 4888154, 2021, doi: 10.1155/2021/4888154.

[9] T. M. Shami et al., "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 10031-10061, 2022, doi: 10.1109/ACCESS.2022.3142859.

[10] J. Bi, H. Ma, H. Yuan, and J. Zhang, "Accurate Prediction of Workloads and Resources with Multi-Head Attention and Hybrid LSTM for Cloud Data Centers," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 3, pp. 375-384, 2023, doi: 10.1109/TSUSC.2023.3259522.

[11] Y. Xing, "Work Scheduling in Cloud Network Based on Deep Q-LSTM Models for Efficient Resource Utilization," *J. Grid Comput.*, vol. 22, Art. no. 36, 2024, doi: 10.1007/s10723-024-09746-6.

[12] M. T. Islam, S. Karunasekera, and R. Buyya, "Performance and Cost-Efficient Spark Job Scheduling Based on Deep Reinforcement Learning in Cloud Computing Environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1695-1710, 2022, doi: 10.1109/TPDS.2021.3124670.

[13] A. Katal, S. Dahiya, and T. Choudhury, "Energy Efficiency in Cloud Computing Data Centers: A Survey on Software Technologies," *Clust. Comput.*, vol. 26, no. 3, pp. 1845-1875, 2023, doi: 10.1007/s10586-022-03713-0.

[14] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "Security-Aware Task Scheduling with Deadline Constraints on Heterogeneous Hybrid Clouds," *J. Parallel Distrib. Comput.*, vol. 153, pp. 15-28, 2021, doi: 10.1016/j.jpdc.2021.03.003.

[15] S. Ouhame, Y. Hadi, and A. Ullah, "An Efficient Forecasting Approach for Resource Utilization in Cloud Data Center Using CNN-LSTM Model," *Neural Comput. Appl.*, vol. 33, no. 16, pp. 10043-10055, 2021, doi: 10.1007/s00521-021-05770-9.

[16] J. Dogani, F. Khunjush, M. R. Mahmoudi, and M. Seydali, "Multivariate Workload and Resource Prediction in Cloud Computing Using CNN and GRU by Attention Mechanism," *J. Supercomput.*, vol. 79, no. 3, pp. 3437-3470, 2023, doi: 10.1007/s11227-022-04782-z.

[17] A. Moazeni, R. Khorsand, and M. Ramezanpour, "Dynamic Resource Allocation Using an Adaptive Multi-Objective Teaching-Learning Based Optimization Algorithm in Cloud," *IEEE Access*, vol. 11, pp. 23407-23419, 2023, doi: 10.1109/ACCESS.2023.3247639.

[18] Y. Alahmad, T. Daradkeh, and A. Agarwal, "Proactive Failure-Aware Task Scheduling Framework for Cloud Computing," *IEEE Access*, vol. 9, pp. 106152-106168, 2021, doi: 10.1109/ACCESS.2021.3101147.

[19] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, "Machine Learning (ML)-Centric Resource Management in Cloud Computing: A Review and Future Directions," *J. Netw. Comput. Appl.*, vol. 204, Art. no. 103405, 2022, doi: 10.1016/j.jnca.2022.103405.

[20] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23-50, 2011, doi: 10.1002/spe.995.